Ministère de l'Emmerdement Supérieur et du temps qui passe
Département du Génie Électrique et Informatique

Bureau d'Étude Systèmes Embarqués Distribués

Métropolitain

version 2009–2010

Présentation du BE 1

On veut concevoir une rame de métro révolutionnaire puisque les voitures ne seraient pas liées mécaniquement entre–elles ! Les avantages sont multiples : la réduction du coût de fabrication, la flexibilité d'utilisation lors de l'ajout de voitures à la rame.

On envisage que chaque voiture possède son propre calculateur relié aux autres par un bus CAN (passant par les rails) permettant d'échanger les informations entre voitures afin d'éviter toute collision.

Il s'agit alors de commander le moteur de chaque voiture de manière à éviter les collisions et à poursuivre une trajectoire permettant de relier les différentes stations dans les temps impartis.

Votre BE consiste à réaliser les logiciels embarqués sur un prototype miniature de la future rame. Dans ce prototype les voitures sont matérialisées par des modèle réduits sur-équipés par nos soins. Le bus CAN est filaire et les calculateurs sont de redoutables STM32F103RB (coeur Cortex-M3).

La fig. 1.1 représente le diagramme d'utilisation des différentes entités logicielles.

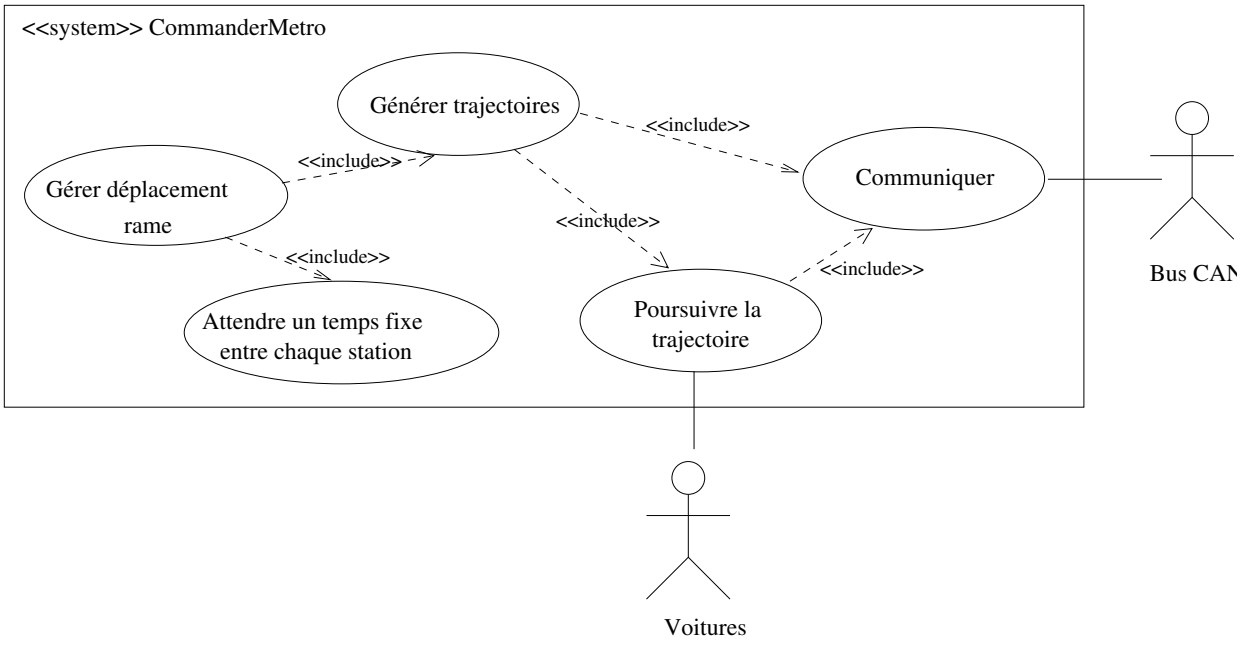

FIGURE 1.1 – Cas d'utilisation

La fig. 1.2 représente l'architecture du prototype montrant l'emplacement des différents calculateurs sur les voitures. Chaque voiture possède sont calculateur C167 (carte MCB167NET) qui est relié aux autres par un bus CAN commun (câbles série sur les connecteur CAN des cartes MCB167NET). Une calculateur supplémentaire est reliée au bus CAN afin d'accueillir le générateur de trajectoires et le point d'accès de débogage.

Plusieurs projets ont déjà été menés à terme :
- la conception mécanique des prototypes ;
- la chaîne d'acquisition de position et de vitesse ainsi que la commande de puissance du moteur ;
- les *driveurs* logiciels du moteur et des capteurs ;
- la modélisation et l'identification du système d'une voiture ;
- un outil de transfert de données entre le C167 et Matlab pour le débugage.

Le schéma technique de la figure 1.3 concerne uniquement une voiture de la rame.

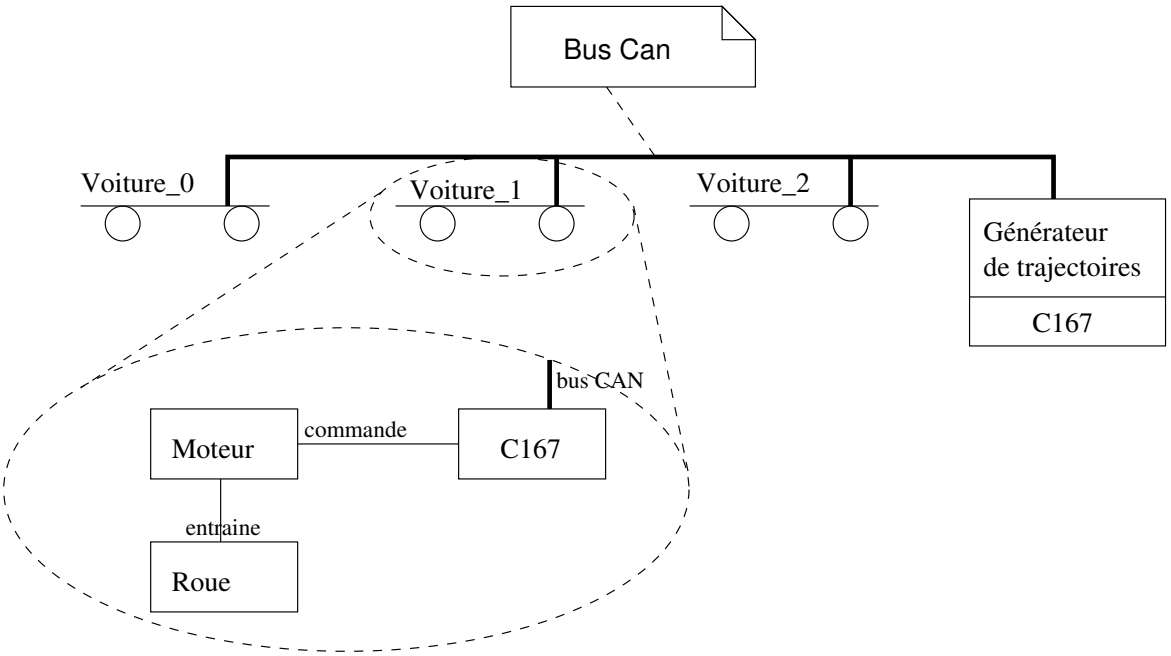

FIGURE 1.2 – Architecture du projet

1.1 Actionneur

De manière très similaire à une véritable voiture de métro, le moteur est commandé par une première boucle de régulation de courant (permettant de le limiter). Le C167 en fonction de ses mesures envoie la consigne de courant de manière à asservir la position ou la vitesse de la voiture.

Le C167 envoie une consigne de courant au format PWM à la carte de régulation de courant :

- un rapport cyclique de 0% correspond à un courant négatif maximal (-10A) ;
- un rapport cyclique de 100% correspond à un courant maximal de 10A ;
- un rapport cyclique de 50% commande un courant nul.

1.2 Capteurs

Vitesse – une génératrice tachymétrique permet de mesurer la vitesse de rotation du moteur et donc la vitesse de la voiture. La tension de la génératrice varie entre +10V et -10V lorsque la vitesse varie entre +3,33 m/s et -3,33 m/s. La carte d'interface permet de ramener cette gamme de tension entre 0 et 5 Volts pour être convertie par un ADC du C167.

Position – la position est mesurée à l'aide de codeurs incrémentaux. Il s'agit de deux capteurs optiques fixes détectant la présence de bandes blanches ou noires (stries) fixées sur la roue. Ses deux capteurs fournissent deux signaux binaires, nommés canal A et canal B, de forme carrée lorsque la roue tourne à vitesse constante. En décalant la bande de stries du canal A par rapport au canal B on peut détecter le sens de rotation de la roue et donc compter ou décompter le nombre d'impulsions reçues sur les canaux. On obtient ainsi une mesure de la position de la roue relative à la position initiale.

Les canaux A et B sont connectés à l'unitée CAPCOM du C167 de manière à générer des interruptions à chaque changement d'état d'un canal.

Consignes – la consigne de position/vitesse est reçue sur le C167 par la liaison du bus CAN. Le bus CAN permet aussi d'accéder aux mesures de position et vitesse des autres voitures de manière à éviter les collisions.

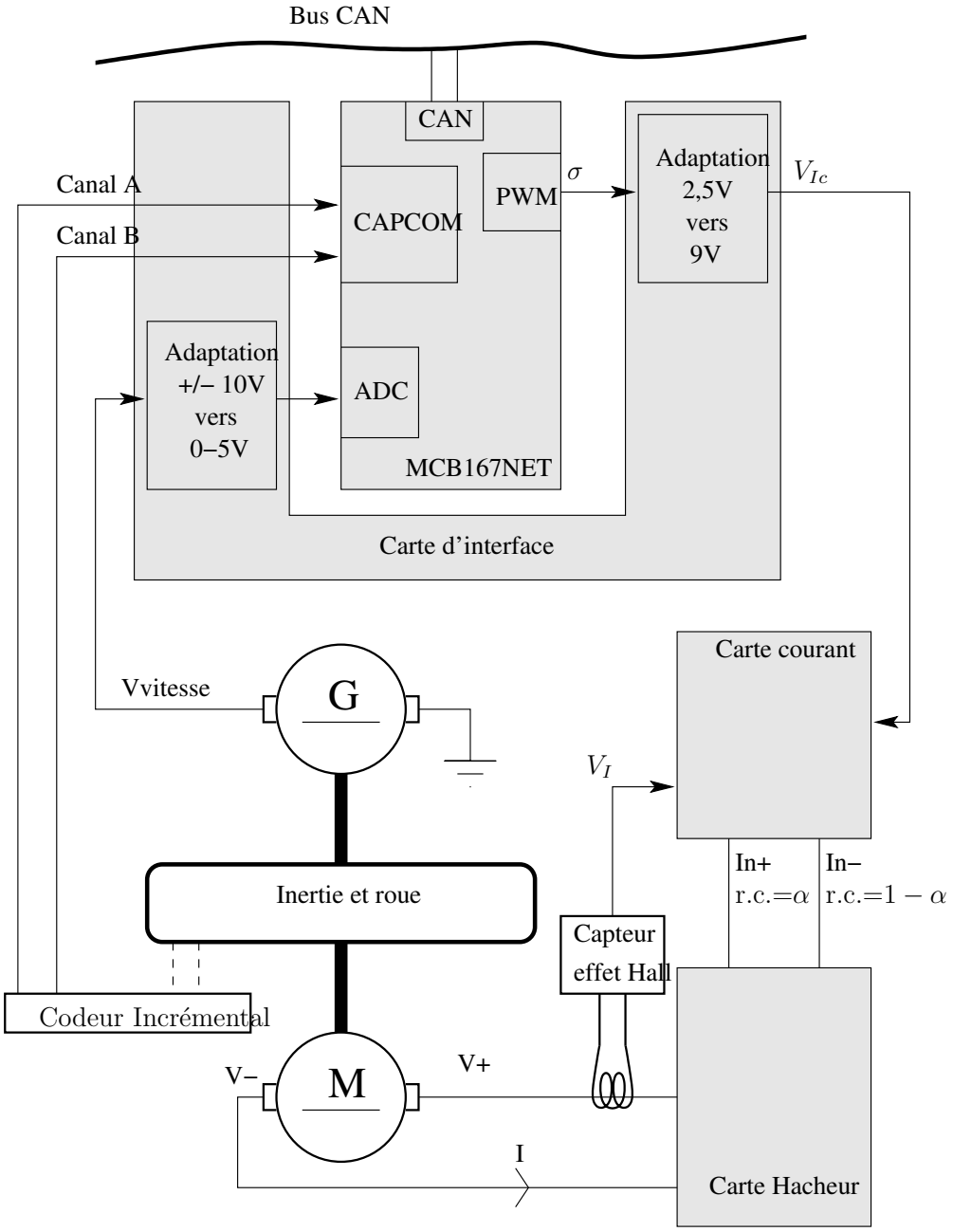

FIGURE 1.3 – Schéma technique de la maquette (trottinette) d'une voiture de la rame

1.3 Librairie de périphériques

Une librairie vous est livrée « clef en main » `driveurs_2008a.c` et `driveurs_2008a.h` qui permet de gérer ces périphériques [1]. Le fichier d'entête contient toutes les informations quand à l'utilisation de cette librairie dont voici les fonctions principales.

1.3.1 Acquisition de la position – TIMER 3 –

Le timer 3 en mode *incremental interface mode* permet de mesurer la position angulaire relative d'une roues.

1. Merci à Thierry Rocacher pour ces bibliothèques de périphériques

Les fonctions associées sont :

void Init_Position(int Position) initialise la position relative mesurée en nombre de pas (largeur d'un strie du codeur incrémental de 4,254 mm environ ;-) ;

int Lire_Position(void) donne la position relative en nombre de pas.

1.3.2 Acquisition de la vitesse – ADC –

L'unité chargée d'acquérir la vitesse consiste à lancer la conversion analogique numérique lorsque une demande de mesure est effectuée

Les fonctions associées sont :

signed int Lire_Vitesse() renvoie un entier signé entre -512 et +511 proportionnel à la vitesse.

float Lire_Vitesse_float() renvoie un flottant entre -3.3 et +3.3 qui représente la vitesse en $m.s^{-1}$.

1.3.3 Commande du courant – PWM –

La consigne de courant envoyée au régulateur de courant est un signal PWM.

Les fonctions associées sont :

void Fixe_Rapport(float Commande_Courant) fixe le rapport cyclique de l'unité PWM à la valeur `Commande_Courant` au format flottant $\in [-1.0; +1.0]$; l'annulation du courant peut être effectuée par l'appel de la fonction `Fixe_Rapport(0.0)`.

Cahier des charges 2

Vos travaux concerneront uniquement le prototype miniature de la rame de métro dénommé dans la suite *métrot-tinette*. Vous vous engagez à :

- §**??** – développer la librairie de communication sur le bus CAN ;
- §2.2 – déterminer les paramètres de la trajectoire du métro entre deux stations ;
- §**??** – développer un simulateur (Matlab/Simulink) de la rame complète et déterminer un retour d'état global permettant de suivre une trajectoire donnée sans collision ;
- §**??** – spécifier/développer/tester le logiciel générique embarqué sur chaque voiture, ainsi que le logiciel spécifique au générateur de trajectoires incluant une liaison de déboguage avec matlab.

2.1 Architecture logicielle

Au cours de ce bureau d'étude, la rame sera composée de trois voitures. Vous devrez cependant prendre en compte le fait que ce nombre pourra être modifié. Pour cela la clef `NB_VOITURES = n` doit être définie pour indiquer le nombre n de voitures dans la rame.

Le logiciel final sera un projet Keil incluant les fichiers et les librairies. Ce projet permettra de compiler et d'exécuter n'importe quelle cible en définissant uniquement la valeur de la clef de compilation qui lui correspond :

- `ID_VOITURE = x` indique qu'il faut générer le logiciel embarqué sur la voiture numéro $x \in 1..n$;
- `CONTROLEUR` indique qu'il faut générer le code du générateur de trajectoire/débogueur.

Les clefs de compilation seront définies dans la partie *define* de l'onglet **C166** de configuration du compilateur.

2.2 Profil de trajectoire

Le générateur de trajectoires donne les consignes de vitesse $v_c(t)$ et de position $x_c(t)$, évidemment l'une est la dérivée de l'autre $v(t) = \dot{x}(t)$.

La trajectoire débutant à l'instant t_0 et se terminant à l'instant t_F doit respecter les contraintes suivante :

- la position finale $x(t_F)$ doit se trouver à une distance L de la position initiale $x(t_0)$ correspondant à la distance entre chaque station de métro ;
- la vitesse de la rame doit être limitée à $V_{max} = 3.33\ m/s$;
- l'accélération de la rame ne doit jamais dépasser en valeur absolue l'accélération maximale $|\dot{v}(t)| < \gamma_{max}$

La figure 2.1 montre le profil d'accélération, le profil de vitesse et de position qui en découlent. La trajectoire est générée à partir d'un profil d'accélération en trois phases : accélération de durée $T_a = 2s$, vitesse constante de croisière, décélération de durée $T_d = Ta = 2s$.

Pour chaque voyage vers une station de métro située à une distance L, le générateur de trajectoire doit planifier le profil de trajectoire adapté et envoyer périodiquement les consignes (toutes les 20 ms).

2.2.1 Délivrables

Calculez les expressions analytiques des paramètres du profil de trajectoire de la figure 2.1 connaissant la distance L à parcourir, la vitesse de croisière V_{max}, et les temps d'accélération/décélération $T_a = T_d$

Ce calcul devra être réalisé en temps discret (période de $T_c = 20ms$) par le calculateur comme l'indique la fig. 2.2.

Vous devrez fournir un fichier *simulink* modélisant le calcul en temps discret du profil de trajectoire.

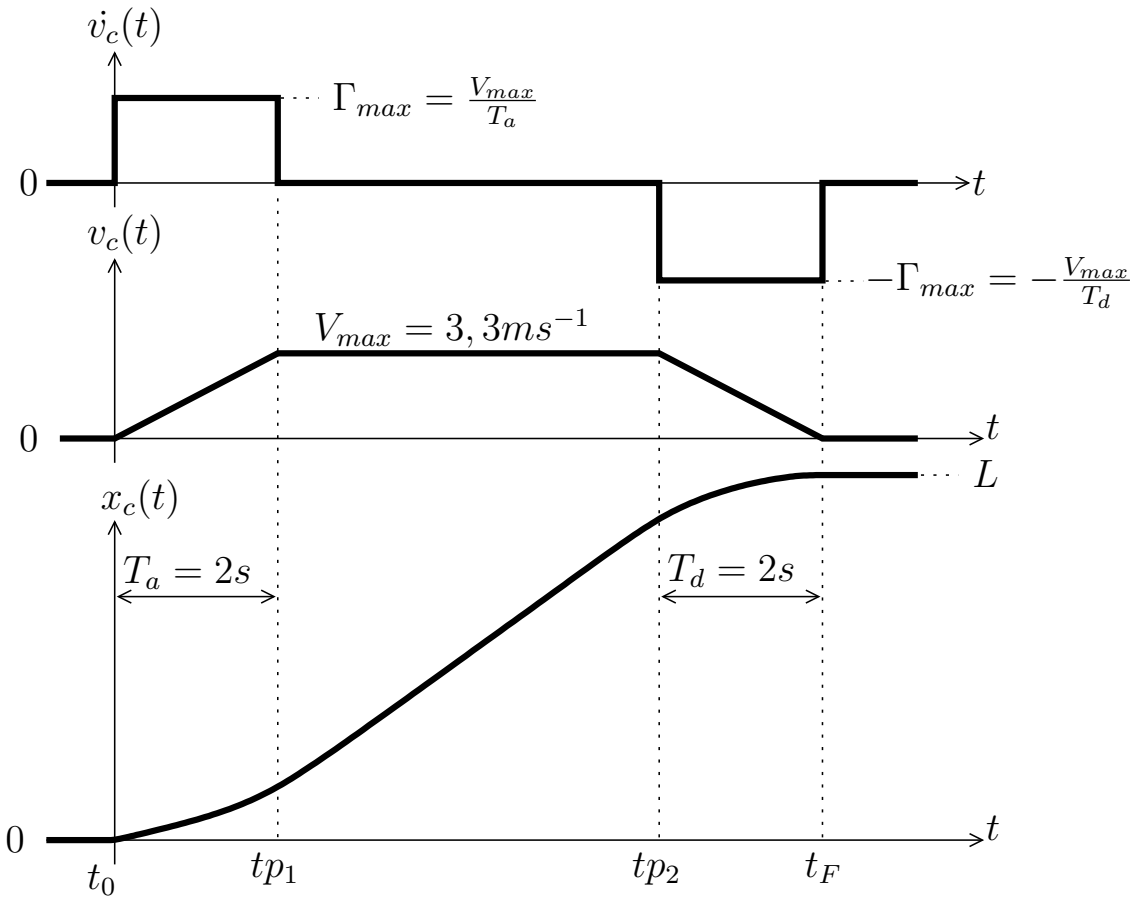

FIGURE 2.1 – Profil d'une trajectoire de consigne

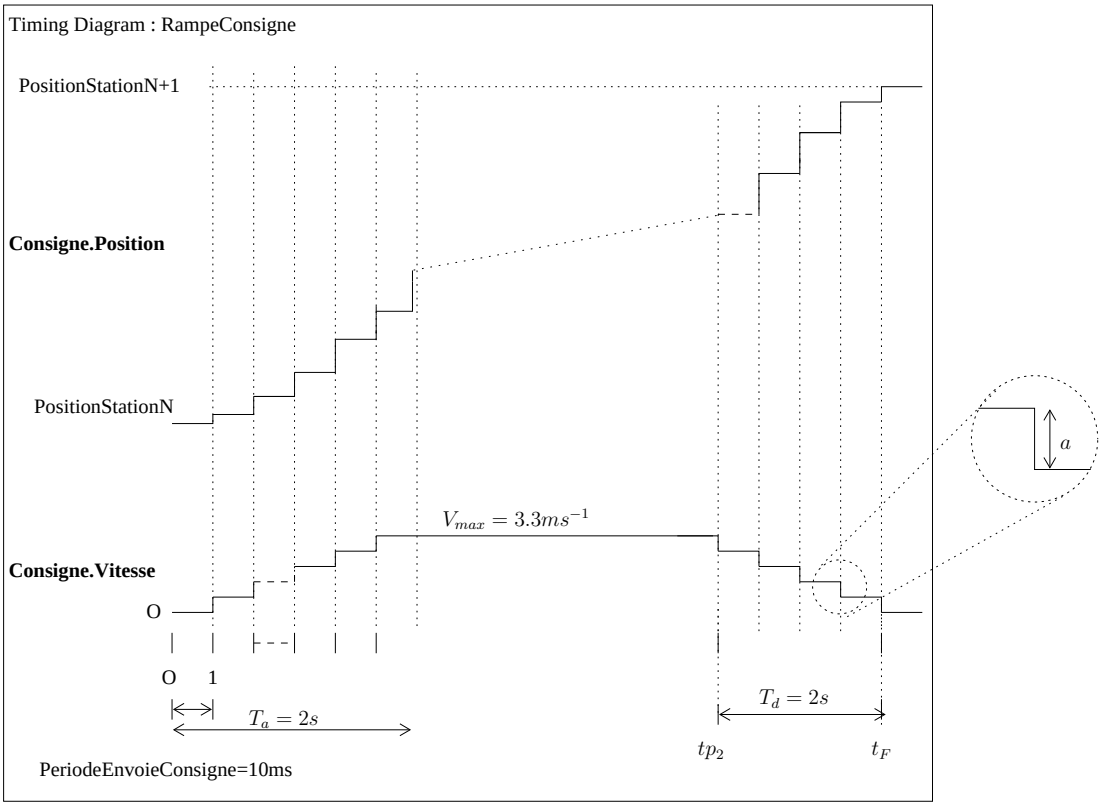

FIGURE 2.2 – Le profil de trajectoire discrétisé

Modèle d'une voiture 3

La commande de puissance du moteur est réalisée par une structure en cascade à double boucle :
- une première boucle actionne le hacheur de manière à asservir le courant dans l'induit du moteur, sa dynamique est rapide puisque liée aux constantes de temps électriques du moteur ;
- une deuxième boucle commande le courant de consigne de la première de manière à asservir la trajectoire de la voiture, sa dynamique est plus lente car liée aux constantes de temps mécanique de la voiture.

La boucle de courant est déjà assurée par un régulateur PI analogique (carte courant). La dernière boucle est à concevoir par vos soins et sera implantée de manière numérique dans le C167.

Cette structure de boucle en cascade est très utilisée en commande de puissance puisqu'elle permet de limiter le courant d'induit en limitant l'amplitude de la consigne de courant. Dans notre cas le courant est limité à $\pm 10A$.

Nous cherchons à modéliser le système « vu par le micro–contrôleur » comme représenté dans la fig. 3.1.

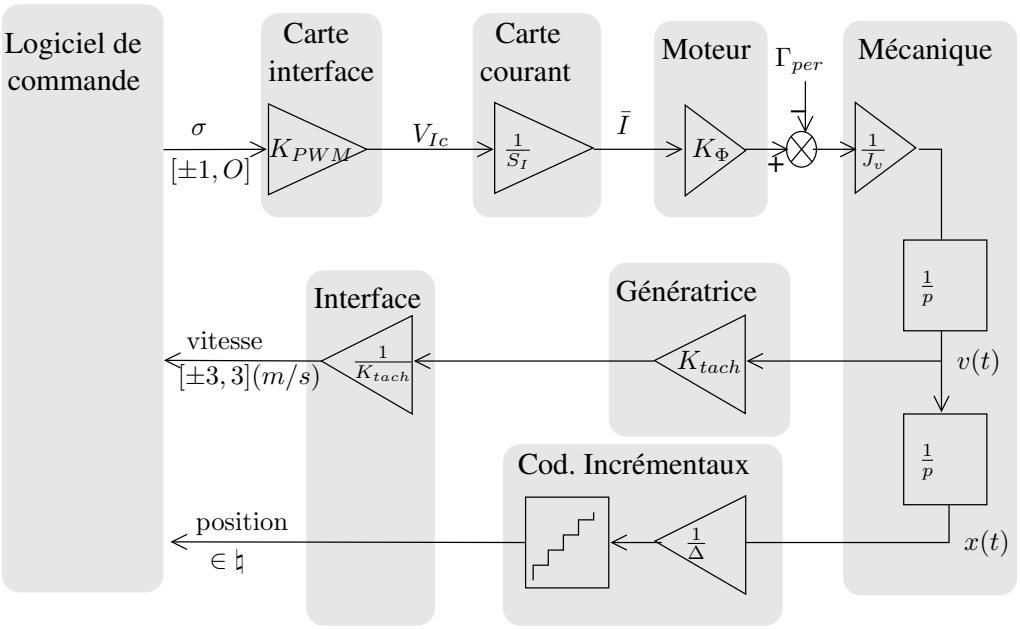

FIGURE 3.1 – Modèle d'une voiture dans la bande 0–300Hz ($K_{PWM} = 5V$, $S_I \approx 0,55\ A/V$, $\frac{K_\Phi}{J_v} \approx 22\ m/A/s^2$, $K_{tach} = \frac{2,5}{3,33}\ V.s/m$, $\Delta = 4,254\ mm$)

3.1 Identification

Une série d'échelon de commande de courant a été envoyée et les mesures de vitesse (avec *signed int lire_vitesse()*) et de position (avec *int lire_position()*) ont été enregistrée sur le C167 puis renvoyées vers Matlab.

Une première identification donne la fonction de transfert en vitesse entre la commande de rapport cyclique $\sigma \in [-1 \quad 1]$ et la vitesse mesurée $v \in [-512 \quad 511]$ suivante :

$$G(p) = \frac{v(p)}{\sigma(p)} = \frac{K}{1+\tau p} \tag{3.1}$$

$K = 485$ à 10%

$\tau = 40$ à 100 ms

Le gain et la constante de temps peuvent varier à cause de l'incertitude due à la charge d'inetrie : rame vide ou rame pleine d'usager.

Une deuxième identification permet d'obtenir le gain d'intégration entre la vitesse mesurée v et la position mesurée x :

$$\begin{array}{l} \dfrac{x(p)}{v(p)} = \dfrac{K_D}{p} \\ K_D = 1.919 \end{array} \tag{3.2}$$

Cette valeur est très précise car elle dépend uniquement de la sensibilité des capteurs.

3.2 Modèle d'état

Dans la suite nous allons modéliser l'ensemble des voitures d'une rame. De manière à traiter ce problème de commande multivariable il faut écrire le modèle dans l'espace d'état.

On peut ainsi représenter le système de la i$^{\text{ème}}$ voiture d'une rame par l'équation d'état :

$$\left\{ \begin{array}{l} \dot{X}_i = A_i\, X_i + B_i\, u_i + G_i\, w_i \\ Y_i = C_i\, X_i \end{array} \right. \tag{3.3}$$

Dans ce cas l'entrée u_i du système est la commande de courant σ, l'entrée de perturbation w_i est le courant I_{per} équivalent à la perturbation Γ_{per}. Le vecteur d'état sera $X_i(t) = [v(t) \quad x(t)]^T$, et la sortie sera $Y = [vitesse(t) \quad position(t)]^T$

Ainsi on peut facilement représenter le modèle d'une rame de métro composée de N voitures en concaténant les modèles avec une représentation d'état « augmentée » :

$$\left\{ \begin{array}{llllllll} \dot{X} & = A^{\frac{2N}{\square}|_{2N}} & X & +B^{\frac{N}{\square}|_{2N}} & U & +G^{\frac{N}{\square}|_{2N}} & W \\ Y & = C^{\frac{2N}{\square}|_{2N}} & X & & & & \end{array} \right. \tag{3.4}$$

Le vecteur d'état est augmenté et devient $X = [X_1 \dots X_N]^T$, le vecteur des entrées devient $U = [u_1 \dots u_N]^T$, celui des perturbations $W = [w_1 \dots w_N]^T$ et le vecteur des sorties sera $Y = [Y_1 \dots Y_N]^T$.

Il est alors possible d'appliquer des méthodes de conception dans l'espaces d'état à ce système multivariable (retour d'état, observateurs, commande optimale).